

Lasserre Lower Bounds and Definability of Semidefinite Programming

Anuj Dawar and Pengming Wang

University of Cambridge Computer Laboratory
`{anuj.dawar, pengming.wang}@cl.cam.ac.uk`

September 27, 2016

Abstract

For a large class of optimization problems, namely those that can be expressed as finite-valued constraint satisfaction problems (VCSPs), we establish a dichotomy on the number of levels of the Lasserre hierarchy of semi-definite programs (SDPs) that are required to solve the problem exactly. In particular, we show that if a finite-valued constraint problem is not solved exactly by its basic linear programming relaxation, it is also not solved exactly by any sub-linear number of levels of the Lasserre hierarchy. The lower bounds are established through logical undefinability results. We show that the linear programming relaxation of the problem, as well as the SDP corresponding to any fixed level of the Lasserre hierarchy is interpretable in a VCSP instance by means of formulas of fixed-point logic with counting. We also show that the solution of an SDP can be expressed in this logic. Together, these results give a way of translating lower bounds on the number of variables required in counting logic to express a VCSP into lower bounds on the number of levels required in the Lasserre hierarchy to eliminate the integrality gap.

As a special case, we obtain the same dichotomy for the class of MAXCSP problems, generalizing some earlier Lasserre lower bound results from [18].

1 Introduction

Many natural optimization problems can be expressed as 0–1 integer programming problems. Indeed, since the problem of determining an optimal solution to a 0–1 integer programming problem is NP-complete, in principle any problem in NP can be so expressed in this framework. Any integer programming problem admits a linear programming relaxation obtained by dropping the integrality constraints. This relaxed linear program can then be solved by standard polynomial-time algorithms, however it admits solutions that are not solutions

to the original integer program. The gap between the optimal solution to the integer program and its linear programming relaxation is known as an *integrality gap*. There are various ways that the linear programming relaxation may be tightened by additional constraints to more closely correspond to the original problem. Several systematic ways have been studied in the literature of constructing hierarchies of ever tighter linear or semidefinite programs, including those of Sherali-Adams [19], Lovasz-Schrijver [15] and Lasserre [13]. Of these, the Lasserre hierarchy is the strongest and gives, for each t , a semidefinite program of size $n^{O(t)}$ (where n is the size of the original integer program) that defines a feasible region whose projection on to the original variables includes the solutions of the integer program. When $t = n$, this projection is exactly the convex hull of the solutions to the original integer program. When this can be achieved for smaller values of t , we get substantially faster algorithms for solving (possibly approximately) the original problem. For many combinatorial optimization problems, the Lasserre relaxations provide the best known approximation algorithms (see [7]).

Our aim in this paper is to establish integrality lower bounds, i.e. to establish for particular combinatorial optimization problems P , expressed as an integer programming problem, a lower bound on the value of t such that the t -th level of the Lasserre hierarchy yields the convex hull of the feasible region of P . Schoenebeck [18] established such lower bounds for the Lasserre hierarchy, showing a linear lower bound on t for a variety of Boolean constraint satisfaction problems, including Max- k -XORSAT. We show that those lower bounds are part of a general pattern. Indeed, we demonstrate a dichotomy on the minimum value of t needed in the Lasserre hierarchy to establish exact solutions to optimization problems in the framework of *finite-valued constraint satisfaction problems* (VCSPs). That is, any such VCSP is either already solved by simply relaxing the integrality constraints in its 0–1 linear program formulation (resulting in the so-called *basic linear program relaxation* (BLP)), or requires a linear number of Lasserre relaxation steps to be solved exactly. As a direct consequence, we obtain the same dichotomy for the class of (weighted) MAXCSP problems.

The study of the complexity of VCSPs has been quite successful in the recent past, culminating in the dichotomy result by Thapper and Živný [20]. There, a complete characterization of the tractable and intractable cases of VCSPs are shown. Namely, any VCSP is either solved exactly by its BLP; or the problem MAXCUT reduces to it, and it is NP-hard. Our main result complements this dichotomy by showing a linear lower bound for the levels of Lasserre relaxations required to exactly solve the hard cases. This characterization of VCSPs into those solvable by their BLP, and those to which MAXCUT reduces, also applies in the context of logical definability. It is known from [9] that in the former case, the class is definable in fixed-point logic with counting (FPC), while in the latter case there is no constant k such that it is definable using only k variables, even in an infinitary logic with counting.

In the present paper we establish a more fine-grained view on the above undefinability result by lifting a previous undefinability result on classical CSPs

from [5], and connect it to the number of levels of the Lasserre hierarchy needed to exactly capture the feasible region of a VCSP. To be precise, we show that if k variables are required to define the VCSP in logic with counting, then $\Omega(k)$ levels of the Lasserre hierarchy are needed to capture the corresponding feasible region. Our result is established in two significant steps. On the one hand, for each k , there is an FPC interpretation that constructs from an integer program the k -th Lasserre relaxation of that program by extending methods of [9]. On the other hand, we show, using methods of [3, 4], that there is an FPC interpretation that can, given an explicitly given semidefinite program, define its optimal solution, up to a given approximation. The dichotomy is then completed by showing that, for every VCSP that is not captured by the basic linear program relaxation, there is a linear lower bound on the number of variables required to define it.

We begin by introducing the necessary definitions from combinatorial optimization, logic and constraint satisfaction problems in Section 2. Section 3 formulates the main result and the steps establishing are then given in Sections 4, 5, and 6.

2 Background

Notation. We write \mathbb{N} for the natural numbers, \mathbb{Z} for the integers, \mathbb{Q} for the rational numbers, and use a superscript plus to denote the non-negative subset, e.g. $\mathbb{Z}^+ = \mathbb{N} \cup \{0\}$.

Given sets A and I , an A -valued *vector* v indexed by I is a function $v : I \rightarrow A$. Often we simply use the subscript notation, writing v_i for $v(i)$. When there is no explicit index set given, vectors are indexed by an initial segment of the natural numbers: $\{1, \dots, d\}$. A *matrix* M is a vector which is indexed by a product set $I \times J$. We use $M_{i,j}$ to denote $M(i, j)$. We write M^T to denote the *transposed* matrix of M , defined as $M_{i,j}^T := M_{j,i}$. A matrix is called *symmetric* if $M = M^T$.

For a rational valued vector $v \in \mathbb{Q}^I$ over some index set I , its *norm* $\|v\|$ is defined as the L^2 -norm over \mathbb{Q}^I , that is, $\|v\| := \sqrt{\sum_{i \in I} v_i^2}$. The *inner product* of two vectors $a, b \in \mathbb{Q}^I$ is defined as $\langle a, b \rangle := \sum_{i \in I} a_i b_i$. We also occasionally write this as a matrix multiplication of vectors $a^T b$. In the case of matrices, this definition of $\langle \cdot, \cdot \rangle$ coincides with the trace inner product of matrices. Note that the norm of a matrix M is defined as the norm of M seen as a vector. For a set $\mathcal{F} \subseteq \mathbb{Q}^I$ and a vector $v \in \mathbb{Q}^I$, we define the *distance* $d(v, \mathcal{F})$ in the usual way, i.e. as $d(v, \mathcal{F}) := \min_{x \in \mathcal{F}} \|x - v\|$. The *ball* $\mathcal{B}(v, r)$ around v with radius $r \in \mathbb{Q}$ is then the set $\mathcal{B}(v, r) := \{x \in \mathbb{Q}^I \mid \|x - v\| \leq r\}$.

A rational valued symmetric matrix $M \in \mathbb{Q}^{I \times I}$ is *positive semidefinite*, if for any $x \in \mathbb{Q}^I$, it holds $x^T M x \geq 0$. We use $M \succeq 0$ to denote that M is positive semidefinite.

We use the notion of *polynomial time* solvable as used in [11]. That is, for problems where we expect an exact solution, this means that there exists an

algorithm running in time polynomial to the encoding of the input that returns an exact solution to the problem. For a problem with a given error parameter $\delta > 0$, we say it is polynomial time solvable if there is an algorithm running in time polynomial of the encoding of the input and $\log(1/\delta)$, that solves the problem up to the specified error.

2.1 Constraint Satisfaction Programs

The class of constraint satisfaction problems (CSPs) provides a framework in which many common combinatorial problems can be expressed. Examples include k -colouring, k -satisfiability, solvability of linear equations over a finite field, and many more. Here we consider CSPs that are parameterized by a fixed *domain* and a *constraint language*, and their optimization variant of so called *finite-valued* CSPs. Finite-valued CSPs have been extensively studied in the recent past and generalize common optimization problems, such as the class of MAXCSPs.

Definition 1. A domain D is a finite, non-empty set. A constraint language Γ over D is a set of relations over D , where each $R \in \Gamma$ is a relation of some arity $m = \text{ar}(R)$, and $R \subseteq D^m$.

An instance of the constraint satisfaction problem over (D, Γ) is then a pair $I = (V, C)$, where V is a finite set of variables, and C is a finite set of constraints. Each constraint $c \in C$ is a pair (s, R) associating a relation $R \in \Gamma$ with a scope $s \in V^{\text{ar}(R)}$.

We say an assignment $h : V \rightarrow D$ satisfies a constraint $c = (s, R)$ if $h(s) \in R$. The goal is to decide whether there exists an assignment that satisfies all constraints $c \in C$.

Example 1. Let $D = \{0, 1, 2\}$, and $\Gamma = \{\neq\}$, that is, Γ contains the inequality relation over D . We obtain the 3-colouring problem as $\text{CSP}(D, \Gamma)$: A instance graph $G = (V, E)$ is simply interpreted as a $\text{CSP}(D, \Gamma)$ -instance where the variables are the vertices V , and every edge $(u, v) \in E$ induces the constraint $((u, v), \neq)$.

Instead of just deciding whether an instance is satisfiable or not, we could be interested in *how many* constraints could be satisfied at the same time. This is the optimization problem MAXCSP: We let $\text{MAXCSP}(D, \Gamma)$ be the optimization problem of determining the maximal number of constraints that can be satisfied in a given instance of $\text{CSP}(D, \Gamma)$.

This can be further generalized. In MAXCSPs, every constraint itself is either satisfied or not. If we now allow constraints to be satisfied to *different degrees*, we obtain the framework of *finite-valued CSPs* (VCSPs). This is the framework we will be working in.

Here, a constraint language consists of a set of functions, instead of relations, where a m -ary function $f : D^m \rightarrow \mathbb{Z}^+$ assigns each m -tuple of the domain an integer value. A constraint then associates some tuple of variables with such a function, and we are interested in the maximum value that can be achieved by any assignment. This is formalized in the following definition.

Definition 2. Let D be a domain. A finite-valued constraint language Γ is a set of functions, where each $f \in \Gamma$ has some arity $m = \text{ar}(f)$, and $f : D^m \rightarrow \mathbb{Z}^+$.

An instance of the valued constraint satisfaction problem over (D, Γ) is a pair $I = (V, C)$, where V is a finite set of variables, and C is a finite set of constraints. Each constraint $c \in C$ is a triple (s, f, w) associating a relation $f \in \Gamma$ with a scope $s \in V^{\text{ar}(f)}$ and a weight $w \in \mathbb{Z}^+$.

The value of an assignment $h : V \rightarrow D$ for an instance I is given as $\text{Val}_I(h) := \sum_{(s, f, w) \in C} w \cdot f(h(s))$. The goal is to determine the maximum value $\text{Opt}(I) = \max_h \text{Val}_I(h)$.

Example 2. Let MAXCUT be the problem of determining the value of the maximum cut in a graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{Z}^+$. Furthermore, we fix $D = \{0, 1\}$, and $\Gamma = \{f\}$ where $f(x, y) = 0$ if $x = y$, and $f(x, y) = 1$ if $x \neq y$. An instance of MAXCUT can be interpreted as an instance I of $\text{VCSP}(D, \Gamma)$: The variables are the vertices V , and we have the constraint $((u, v), f, w(u, v))$ for every edge $(u, v) \in E$. The value of the maximum cut is exactly $\text{Opt}(I)$.

Example 3. It is not difficult to see that every $\text{MAXCSP}(D, \Gamma)$ is a finite-valued CSP. For any m -ary relation R , define a function $f_R : D^m \rightarrow \{0, 1\}$ as $f(t) = 1$ if $t \in R$ and $f(t) = 0$ if $t \notin R$. Let Γ' be the finite-valued constraint language that consist of f_R for all $R \in \Gamma$. Then, $\text{MAXCSP}(D, \Gamma) = \text{VCSP}(D, \Gamma')$.

When talking complexity classes and reductions, it is often more convenient to also phrase VCSPs as decision problems. Abusing notation, we will use $\text{VCSP}(D, \Gamma)$ to also denote the set of pairs (I, t) such that I is an instance with $\text{Opt}(I) \geq t$. The decision problem is then, given any pair (I, t) , to decide whether $(I, t) \in \text{VCSP}(D, \Gamma)$.

In the study of valued constraint satisfaction problems, linear programming in particular has proven to be a useful tool. In fact, every instance of $\text{VCSP}(D, \Gamma)$ is equivalent to the following *integer* linear program.

For an instance $I = (V, C)$, the program contains variables $\lambda_{c,x}$ for every $c \in C$ with $c = (s, f, w)$ and $x \in D^{\text{ar}(s)}$, and $\mu_{v,a}$ for every $v \in V$ and $a \in D$. A solution that sets a variable $\lambda_{c,x}$ to 1 then corresponds to an assignment that assigns the scope of the constraint c to the tuple x . In order to maintain consistency of the assignment between constraints, the variable $\mu_{v,a}$ encodes whether the variable v is assigned the value a . The objective is then to maximize the value of the assignment. The 0–1 program is then given below.

$$\begin{aligned}
\max \sum_{c \in C} \sum_{x \in D^{\text{ar}(s)}} \lambda_{c,x} \cdot w \cdot f(x) \quad & \text{where } c = (s, f, w), \text{ s.t.} \\
\sum_{x \in D^{\text{ar}(s)}; x_i = a} \lambda_{c,x} &= \mu_{s_i, a} & \forall c \in C, a \in D, i \in [\text{ar}(s)] \\
\sum_{a \in D} \mu_{v,a} &= 1 & \forall v \in V \\
\lambda_{c,x} &\in \{0, 1\} & \forall c \in C, x \in D^{\text{ar}(s)} \\
\mu_{v,a} &\in \{0, 1\} & \forall v \in V, a \in D
\end{aligned}$$

If we relax the integrality constraints of the above LP to $0 \leq \lambda_{c,x} \leq 1$ and $0 \leq \mu_{v,a} \leq 1$, we obtain the *basic linear program relaxation* $\text{BLP}(I)$. Since this allows rational assignments, this LP can be solved exactly in polynomial time. In general the optimal value of $\text{BLP}(I)$ only gives an overestimate of the optimal value $\text{Opt}(I)$ to the VCSP. However, there are (D, Γ) for which any instance I of $\text{VCSP}(D, \Gamma)$ is solved by $\text{BLP}(I)$ exactly, and solving the LP gives an exact algorithm for $\text{VCSP}(D, \Gamma)$. Thapper and Živný [20] give a complete characterization of those cases – and show that in all other cases the problem is NP-hard.

Theorem 3. *For any domain D , and any finite-valued constraint language Γ , either every instance I of $\text{VCSP}(D, \Gamma)$ is solved by $\text{BLP}(I)$; or the problem MAXCUT polynomial-time reduces to $\text{VCSP}(D, \Gamma)$.*

Our main result expands on this dichotomy result, and shows a linear lower bound of the required levels of the Lasserre hierarchy for all the cases not solved by the BLP relaxation.

2.2 Semidefinite Optimization

We give a brief overview of the basic notions of semidefinite programs.

In general, semidefinite programming refers to a framework of constrained optimization problems where the search space is over the set of *positive semidefinite matrices*. More specifically, in a typical semidefinite program we are interested in the entries of a symmetric matrix $X \in \mathbb{Q}^{V \times V}$ that maximizes the value of an objective function $\langle C, X \rangle$, subject to a set of constraints of the form $\langle A_i, X \rangle \leq b_i$ with the additional constraint that X is a positive semidefinite matrix.

Definition 4. *Let V, M be sets, and let V be non-empty. A semidefinite program (SDP) is given by an objective matrix $C \in \mathbb{Q}^{V \times V}$, a $\mathbb{Q}^{V \times V}$ -valued vector $\mathcal{A} \in \mathbb{Q}^{M \times (V \times V)}$, and a vector $b \in \mathbb{Q}^M$.*

We call $\mathcal{F}_{\mathcal{A}, b} := \{X \in \mathbb{Q}^{V \times V} \mid X \succeq 0, \langle A_i, X \rangle \leq b_i, A_i = \mathcal{A}(i), i \in M\}$ the set of feasible solutions.

We sometimes call sets that can be defined as feasible regions of an SDP a *positive semidefinite set*. This definition covers SDPs that are in the so-called *conic standard form*. Sometimes however it is more convenient to specify SDPs in their *inequality standard form*. In this form, the SDP is instead given by a matrix $Z \in \mathbb{Q}^{M \times M}$, a matrix-valued vector $\mathcal{Y} \in \mathbb{Q}^{V \times (M \times M)}$, and an objective vector $c \in \mathbb{Q}^V$. The feasible region is then defined as $\mathcal{F}_{\mathcal{Y}, Z} := \{x \in \mathbb{Q}^V \mid Z + \sum_{v \in V} x_v \mathcal{Y}_v \succeq 0\}$. The two standard forms can be converted into each other by adding, substituting, and rearranging variables. The number of additional variables needed can be bounded by a linear function in both cases. Hence, we will use whichever representation is most convenient for any given case.

Note that by the definition of inner product, the objective function $\langle C, X \rangle$ is a linear function over the variables $x_{u,v}, u, v \in V$. Likewise, constraints of

the form $\langle A_i, X \rangle \leq b_i$ are also linear inequalities over the entries $x_{u,v}$. Hence, we can view semidefinite programs as a generalization of *linear programs*, with the additional constraint that the solution must define a positive semidefinite matrix. In fact, the semidefinite constraint $X \succeq 0$ essentially imposes an *infinite* set of additional linear constraints, namely $a^T X a \geq 0$ for all $a \in \mathbb{Q}^V$.

The feasible region of a semidefinite program is convex, since it can be described as an intersection of (infinitely many) halfspaces. Classically, in the context of convex optimization, we are interested in the solutions of the two main problems of *optimization* and *separation*. As a technical point, the optimal solution to a semidefinite program, or convex problems in general, is not necessarily rational, so we can only express it up to a finite precision. This gives rise to the weak formulations of the problems where we allow an additive error to be specified in the input.

Definition 5. Let V be a non-empty set. Given a vector $c \in \mathbb{Q}^V$ and a convex set $\mathcal{F} \subseteq \mathbb{Q}^V$, the strong optimization problem is to either find an element $y = \operatorname{argmax}_{x \in \mathcal{F}} \langle c, x \rangle$, or to determine that \mathcal{F} is empty, or that $\max_{x \in \mathcal{F}} \langle c, x \rangle$ is unbounded.

In the weak optimization problem we are given an additional error parameter $\delta > 0$, and want to determine an element y that is δ -close to \mathcal{F} , i.e. $d(y, \mathcal{F}) \leq \delta$, that is also δ -maximal, i.e. $\langle c, y \rangle + \delta \geq \max_{x \in \mathcal{F}} \langle c, x \rangle$, or, again, to determine that $\max_{x \in \mathcal{F}} \langle c, x \rangle$ is unbounded.

Definition 6. Let V be a non-empty set. Given a vector $y \in \mathbb{Q}^V$ and a convex set $\mathcal{F} \subseteq \mathbb{Q}^V$, the strong separation problem is the problem of determining either that $y \in \mathcal{F}$, or finding a vector $s \in \mathbb{Q}^V$ with $\langle s, y \rangle > \max\{\langle s, x \rangle \mid x \in \mathcal{F}\}$ and $\|s\|_\infty = 1$.

In the weak separation problem, we are given an additional parameter $\delta > 0$, and are looking to determine that either y is δ -close to \mathcal{F} , i.e. $d(y, \mathcal{F}) \leq \delta$, or to find a vector $s \in \mathbb{Q}^V$, such that $\langle s, y \rangle + \delta > \max\{\langle s, x \rangle \mid x \in \mathcal{F}\}$ and $\|s\|_\infty = 1$.

The relationship between the optimization and separation problem of a given convex set is well-studied and is most prominently expressed by Grötschel, Lovász, and Shrijver [11] as being polynomial time equivalent. More precisely, with the additional assumptions that the set \mathcal{F} is *full-dimensional* (\mathcal{F} has positive volume in \mathbb{Q}^V) and *bounded* (\mathcal{F} is contained within a ball of finite radius), the weak optimization problem for \mathcal{F} is solvable in polynomial time if, and only if, the corresponding weak separation problem is solvable in polynomial time. The following is essentially the statement of Theorem 4.2.7 in [11].

Theorem 7. Let V be a non-empty set, and $\mathcal{F} \subseteq \mathbb{Q}^V$ a full-dimensional convex set that is located inside the ball $\mathcal{B}(0, R)$ for some known value R . The weak optimization problem on \mathcal{F} is solvable in polynomial time if its weak separation problem is solvable in polynomial time.

Note that in the special case where \mathcal{F} is a rational polyhedron (for instance the feasible region of a linear program), even the strong versions of the problems can be solved in polynomial time, and the additional assumptions just introduced can be avoided.

The main tool in the reduction from optimization to separation is the so-called *ellipsoid method* (see [11]), which is an algorithm that repeatedly calls a blackbox solver for the separation problem, called a *separation oracle*, in order to locate a feasible point. Finding an algorithm for the optimization problem then reduces to finding a suitable separation oracle for its feasible region. In the case of semidefinite programs, a simple (strong) separation oracle is given by Algorithm 1. Note that this algorithm assumes that we can compute the eigenvalues and eigenvectors with infinite precision. Realistically we will have to work with finite precision approximations that can be obtained in polynomial time. For our purposes, in Section 5.1, we will formulate a slightly modified algorithm that serves as a weak separation oracle that can be expressed in fixed-point logic with counting (FPC).

Anderson et al. [3] have shown that the ellipsoid method can be suitably expressed in fixed-point logic with counting, at least in the case of polyhedra. In the present paper we extend their construction to semidefinite sets. Together with the FPC-definable weak separation oracle, this yields our definability result for semidefinite programs.

Algorithm 1 Separation oracle for semidefinite programs

Input: $\mathcal{A} = \{A_1, \dots, A_m \in \mathbb{Q}^{V \times V}\}$, $b \in \mathbb{Q}^m$, $Y \in \mathbb{Q}^{V \times V}$.

Output: Solves separation problem on $\mathcal{F}_{\mathcal{A},b}$ and Y .

- 1: **function** SEPARATION(\mathcal{A}, b, Y):
 - 2: **if** there is $A_i \in \mathcal{A}$ such that $\langle A_i, Y \rangle > b_i$ **then**
 - 3: **return** $\frac{1}{\|A_i\|} A_i$
 - 4: Compute Eigenvalues $\{\lambda_1, \dots, \lambda_{|V|}\}$ of Y
 - 5: **if** there is $\lambda_i < 0$ **then**
 - 6: $v \leftarrow$ Eigenvector corresponding to λ_i
 - 7: **return** $(-1) / \|vv^T\| \cdot vv^T$
 - 8: **return** ACCEPT
-

2.3 Lasserre Hierarchy

One of the most common applications of semidefinite programming is to give approximation algorithms for hard combinatorial problems. For a large class of problems, namely those that can be expressed as integer programs, a generic way to find approximations is to drop the integrality condition so that a rational solution can be efficiently computed. The value of the optimal rational solution serves as an upper bound to the optimal value of the integer problem and can be used as an approximation. The concept of relaxation hierarchies extends this idea further. Instead of solving the basic relaxation, these hierarchies define a sequence of linear or semidefinite programs that provide increasingly finer approximations to the original integer solution. This is achieved by adding, at each level of the hierarchy, additional constraints to the basic relaxation that are preserved under the original integer program, but may cut away rational

solutions not in the convex hull of integer solutions.

A prominent example of such a relaxation hierarchy is the *Lasserre hierarchy* which for a given 0–1 linear program defines a sequence of semidefinite programs.

Definition 8. Let V, M be sets and $\mathcal{K} := \{x \in \mathbb{Q}^V \mid Ax \geq b\}$ a polytope given by $A \in \mathbb{Q}^{U \times V}, b \in \mathbb{Q}^U$.

For a vector $y \in \mathbb{Q}^{\wp(V)}$, and an integer t with $1 \leq t \leq |V|$, we define the t -th moment matrix of y , $M_t(y)$ as the $\wp_t(V) \times \wp_t(V)$ -matrix with entries

$$M_t(y)_{I,J} := y_{I \cup J}, \text{ for } |I|, |J| \leq t.$$

Similarly, the t -th moment matrix of slacks of y, A, b , and some $u \in U$ is given by

$$S_t^u(y)_{I,J} := \sum_{v \in V} A_{u,v} y_{I \cup J \cup \{v\}} - b_u y_{I \cup J}, \text{ for } |I|, |J| \leq t.$$

Finally, the t -th level of the Lasserre hierarchy of \mathcal{K} , $\text{Las}_t(\mathcal{K})$ is the positive semidefinite set defined by

$$\text{Las}_t(\mathcal{K}) := \{y \in \mathbb{Q}^{\wp_{2t+1}(V)} \mid y_\emptyset = 1, M_t(y) \succeq 0, S_t^u(y) \succeq 0 \text{ for all } u \in U\}.$$

We write $\text{Las}_t^\pi(\mathcal{K}) := \{y_{\{v\}}, v \in V \mid y \in \text{Las}_t(\mathcal{K})\}$ for the projection of $\text{Las}_t(\mathcal{K})$ onto the original variables.

The general usage of the Lasserre hierarchy is as follows. Assume we have a 0–1 program where the feasible region is defined as $\mathcal{K} \cap \{0, 1\}^V$. Now, instead of optimizing over the integer region, we can define $\text{Las}_t(\mathcal{K})$ for some level t , and solve the corresponding SDP. For a fixed constant t , this SDP has a polynomial number of new variables, and the optimum can be obtained in polynomial time. This optimum, when projected down onto the original variables, serves as an approximation to the optimum in $\mathcal{K} \cap \{0, 1\}^V$.

The following basic properties of the Lasserre hierarchy establish that $\text{Las}_t(\mathcal{K})$ is indeed a relaxation of $\mathcal{K} \cap \{0, 1\}^V$. We write \mathcal{K}^* for the polytope that is defined by the convex hull of the integer points in \mathcal{K} , i.e. $\mathcal{K}^* := \text{conv}(\mathcal{K} \cap \{0, 1\}^V)$.

Lemma 9. Let $\mathcal{K} = \{x \in \mathbb{Q}^V \mid Ax \geq b\}$, and $y \in \text{Las}_t(\mathcal{K})$ for $t \in \{1, \dots, |V|\}$. Then,

1. $\mathcal{K}^* \subseteq \text{Las}_t^\pi(\mathcal{K})$.
2. $\text{Las}_0(\mathcal{K}) \supseteq \text{Las}_1(\mathcal{K}) \supseteq \dots \supseteq \text{Las}_{|V|}(\mathcal{K})$.
3. $\text{Las}_0^\pi(\mathcal{K}) \subseteq \mathcal{K}$, and $\mathcal{K}^* = \text{Las}_{|V|}^\pi(\mathcal{K})$.

Proof. See for instance in [17] □

Definition 10. Let $I = (c, \mathcal{K})$ be a 0–1 linear program with a objective vector $c \in \mathbb{Q}^V$ optimizing over a feasible region $\mathcal{K} \cap \{0, 1\}^V$. We say that I is captured

at the t^{th} level of the Lasserre hierarchy if $\text{Las}_t^\pi(\mathcal{K}) = \mathcal{K}^*$. We write $l(I)$ for the minimum t , such that I is captured at the t^{th} level.

For a class of 0–1 linear programs C , we say that C is captured at the t^{th} level, if every program in C is captured at the t^{th} level of the Lasserre hierarchy.

We denote by $L_C(n)$ the function that maps an integer n to the lowest level t at which any 0–1 program in C with size n is captured. That is, $L_C(n) := \max_{I \in C; |V| \leq n} l(I)$.

We see that at a sufficiently high level, namely at most at level $t = |V|$, any 0–1 program is captured by the t^{th} level Lasserre relaxation. In those cases, the optimum of the Lasserre set yields not only an approximate optimum, but is the exact optimal value of the original 0–1 problem.

Note that it could in principle occur that some instance $I = (c, \mathcal{K})$ is solved exactly by a Lasserre relaxation of level t strictly less than $l(I)$. That is, \mathcal{K}^* is strictly contained in $\text{Las}_t^\pi(\mathcal{K})$, but they coincide in direction of the objective vector c . We argue that for classes of 0–1 programs for which the objective c can be chosen arbitrarily, $L_C(n)$ still is a right notion for the number of Lasserre levels required to solve instances of C exactly. This is formalized in the lemma below.

Lemma 11. *For any 0–1 linear program $I = (c, \mathcal{K})$, let $f(I)$ denote the minimum t , such that $\max_{x \in \mathcal{K}^*} \langle c, x \rangle = \max_{x \in \text{Las}_t^\pi(\mathcal{K})} \langle c, x \rangle$.*

Furthermore, let C be a class of 0–1 linear programs, such that, if $I = (c, \mathcal{K})$ is an instance of C , then $J = (c', \mathcal{K})$ is also an instance of C for any choice of $c' \in \mathbb{Q}^V$.

Then, $L_C(n) = \max_{I \in C; |V| \leq n} f(I)$.

Proof. Assume the claim is false. Then we can pick an n , such that $L_C(n) > \max_{I \in C; |V| \leq n} f(I)$, and define $k := \max_{I \in C; |V| \leq n} f(I)$. Now pick an instance $I = (c, \mathcal{K})$ that maximizes the right hand side, i.e. $f(I) = k$. In particular, $l(I) > k$, and there is some direction c' where the maxima of the k th Lasserre relaxation and the convex hull of integer solutions do not match, i.e. $\max_{x \in \mathcal{K}^*} \langle c', x \rangle < \max_{x \in \text{Las}_k^\pi(\mathcal{K})} \langle c', x \rangle$. However by assumption the instance $J = (c', \mathcal{K})$ is also in C , and $f(J)$ must be larger than k . This is a contradiction to the definition of k . \square

Note that the class of VCSPs satisfies the condition of the above lemma: If an instance of $\text{VCSP}(D, \Gamma)$ is represented by a 0–1 linear program (c, \mathcal{K}) , then any other cost vector c' can be achieved by changing the weights on the constraints.

Our main result establishes a dichotomy for VCSPs with respect to $L_C(n)$: For every (D, Γ) , $C = \text{VCSP}(D, \Gamma)$, either $L_C(n) = 0$ ($\text{VCSP}(D, \Gamma)$ is solved by the basic linear program relaxation); or $L_C(n) \in \Omega(n)$.

2.4 Logic

We define the logical notions used throughout the paper.

A relational *vocabulary* τ is a finite sequence of relation and constant symbols $(R_1, \dots, R_k, c_1, \dots, c_l)$, where every relation symbol R_i has a fixed arity $a_i \in \mathbb{N}$. A structure $\mathbf{A} = (\text{dom}(\mathbf{A}), R_1^{\mathbf{A}}, \dots, R_k^{\mathbf{A}}, c_1^{\mathbf{A}}, \dots, c_l^{\mathbf{A}})$ over the signature τ (or a τ -*structure*) consists of a non-empty set $\text{dom}(\mathbf{A})$, called the *universe* of \mathbf{A} , together with relations $R_i^{\mathbf{A}} \subseteq \text{dom}(\mathbf{A})^{a_i}$ and constants $c_j^{\mathbf{A}} \in \text{dom}(\mathbf{A})$ for each $1 \leq i \leq k$ and $1 \leq j \leq l$. Members of the set $\text{dom}(\mathbf{A})$ are called the *elements* of \mathbf{A} and we define the *size* of \mathbf{A} to be the cardinality of its universe, often written as $|\mathbf{A}|$.

2.4.1 Fixed-point Logic with Counting

Fixed-point logic with counting (FPC) is an extension of inflationary fixed-point logic with the ability to express the cardinality of definable sets. Here we give a bare-bones definition of the logic. For more details, we refer the reader to [10, 14]. The logic has two sorts of first-order variables: *element variables*, which range over elements of the structure on which a formula is interpreted in the usual way, and *number variables*, which range over some initial segment of the natural numbers. We usually write element variables with lower-case Latin letters x, y, \dots and use lower-case Greek letters μ, η, \dots to denote number variables.

The atomic formulas of $\text{FPC}[\tau]$ are all formulas of the form $\mu = \eta$ or $\mu \leq \eta$, where μ, η are number variables; $s = t$ where s, t are element variables or constant symbols from τ ; and $R(t_1, \dots, t_m)$, where each t_i is either an element variable or a constant symbol and R is a relation symbol (i.e. either a symbol from τ or a relational variable) of arity m . Each relational variable of arity m has an associated type from $\{\text{elem}, \text{num}\}^m$. The set $\text{FPC}[\tau]$ of FPC *formulas* over τ is built up from the atomic formulas by applying an inflationary fixed-point operator $[\text{ifp}_{R, \bar{x}} \phi](\bar{t})$; forming *counting terms* $\#_x \phi$, where ϕ is a formula and x an element variable; forming formulas of the kind $s = t$ and $s \leq t$ where s, t are number variables or counting terms; as well as the standard first-order operations of negation, conjunction, disjunction, universal and existential quantification. Collectively, we refer to element variables and constant symbols as *element terms*, and to number variables and counting terms as *number terms*.

For the semantics, number terms take values in $\{0, \dots, n\}$, where $n = \text{dom}(\mathbf{A})$ and element terms take values in $\text{dom}(\mathbf{A})$. The semantics of atomic formulas, fixed-points and first-order operations are defined as usual (c.f., e.g., [10] for details), with comparison of number terms $\mu \leq \eta$ interpreted by comparing the corresponding integers in $\{0, \dots, n\}$. Finally, consider a counting term of the form $\#_x \phi$, where ϕ is a formula and x an element variable. Here the intended semantics is that $\#_x \phi$ denotes the number (i.e. the element of $\{0, \dots, n\}$) of elements that satisfy the formula ϕ .

Throughout the paper, we make frequent use of the Immerman-Vardi theorem [10], which establishes that fixed-point logic can express all polynomial-time properties of finite ordered structures. It follows that in FPC we can express all polynomial-time relations on the number domain.

We write C^k for the fragment of first-order logic with counting quantifiers

consisting of those formulas that can be written using at most k distinct variables. It is easy to see that any structure with n elements can be described up to isomorphism by a formula using no more than n variables. It follows that any collection of structures, each of which has no more than n elements, can also be characterized up to isomorphism by a formula with no more than n variables.

The minimum number of variables needed to define a class of structures in C^k turns out to be a useful measure of complexity. This motivates the definition of the *counting width* of a class.

Definition 12. *For any class of structures \mathcal{C} , the counting width of \mathcal{C} is the function $\nu_{\mathcal{C}} : \mathbb{N} \rightarrow \mathbb{N}$ where $\nu_{\mathcal{C}}(n)$ is the minimum value k such that there is a formula ϕ in C^k , for which any structure \mathbf{A} with $|\text{dom}(\mathbf{A})| \leq n$, it holds $\mathbf{A} \models \phi \Leftrightarrow \mathbf{A} \in \mathcal{C}$.*

It is clear that $\nu_{\mathcal{C}} = \Omega(n)$ for any class \mathcal{C} . It is known that if \mathcal{C} is definable in FPC, then $\nu_{\mathcal{C}}$ is bounded by a constant (see [16]). The converse is not true in general as there are even undecidable classes \mathcal{C} for which $\nu_{\mathcal{C}}$ is bounded by a constant. However, the converse holds in special cases, such as for constraint satisfaction problems. Here we have a dichotomy: every $\mathcal{C} = \text{CSP}(D, \Gamma)$ is either definable in FPC or has unbounded $\nu_{\mathcal{C}}$. For an explanation see [9] where this result is extended to finite valued CSPs.

In Section 4, we will show that the counting width of finite-valued CSPs is either bounded by a constant, or is $\Omega(n)$. We use this for our main result to establish a similar dichotomy on the number of levels of the Lasserre hierarchy needed to capture the 0–1 linear programs coding instances of VCSPs.

2.4.2 Interpretations

We frequently consider ways of defining one structure within another in some logic L , such as first-order logic or FPC. Consider two signatures σ and τ and a logic L . An *m -ary L -interpretation of τ in σ* is a sequence of formulae of L in vocabulary σ consisting of: (i) a formula $\delta(x)$; (ii) a formula $\varepsilon(x, y)$; (iii) for each relation symbol $R \in \tau$ of arity k , a formula $\phi_R(x_1, \dots, x_k)$; and (iv) for each constant symbol $c \in \tau$, a formula $\gamma_c(x)$, where each x, y or x_i is an m -tuple of free variables. We call m the *width* of the interpretation. We say that an interpretation Θ associates a τ -structure \mathbf{B} to a σ -structure \mathbf{A} if there is a surjective map h from the m -tuples $\{a \in \text{dom}(\mathbf{A})^m \mid \mathbf{A} \models \delta[a]\}$ to \mathbf{B} such that:

- $h(a_1) = h(a_2)$ if, and only if, $\mathbf{A} \models \varepsilon[a_1, a_2]$;
- $R^{\mathbf{B}}(h(a_1), \dots, h(a_k))$ if, and only if, $\mathbf{A} \models \phi_R[a_1, \dots, a_k]$;
- $h(a) = c^{\mathbf{B}}$ if, and only if, $\mathbf{A} \models \gamma_c[a]$.

Note that an interpretation Θ associates a τ -structure with \mathbf{A} only if ε defines an equivalence relation on $\text{dom}(\mathbf{A})^m$ that is a congruence with respect to the relations defined by the formulae ϕ_R and γ_c . In such cases, however, \mathbf{B} is

uniquely defined up to isomorphism and we write $\Theta(\mathbf{A}) := \mathbf{B}$. Throughout this paper, we will often use interpretations where ε is simply defined as the usual equality on a_1 and a_2 . In these instances, we omit the explicit definition of ε .

The notion of interpretations is used to define logical reductions. Let C_1 and C_2 be two classes of σ - and τ -structures respectively. We say that C_1 *L-reduces* to C_2 if there is an L-interpretation Θ of τ in σ , such that $\Theta(\mathbf{A}) \in C_2$ if and only if $\mathbf{A} \in C_1$, and we write $C_1 \leq_L C_2$.

It is not difficult to show that formulas of FPC compose with FPC-reductions in the sense that, given an interpretation Θ of τ in σ and a τ -formula ϕ , we can define a σ -formula ϕ' such that $\mathbf{A} \models \phi'$ if, and only if, $\Theta(\mathbf{A}) \models \phi$. Note that if ϕ uses k variables, the composition ϕ' may contain up to $m \cdot k$ many variables, where m is the width of Θ . Likewise, interpretations themselves compose. That is, given interpretations Θ of τ in σ , and Σ of σ in ρ , we can obtain an interpretation Θ' of τ in ρ by composition: Θ' consists of the functions of Θ where the relation symbols of σ are instead replaced by the corresponding ρ -formulas in Σ .

Finally, dealing with FPC-reductions allows us to track counting width in the following way.

Proposition 13. *Let C_1 and C_2 be two classes of structures, such that $C_1 \leq_{\text{FPC}} C_2$ by some FPC-reduction Θ . Furthermore, let $\theta : \mathbb{N} \rightarrow \mathbb{N}$ be defined as $\theta(n) = \max_{\mathbf{A} \in C_1; |\mathbf{A}| \leq n} |\Theta(\mathbf{A})|$. Then $\nu_{C_1}(n) \in O(\nu_{C_2}(\theta(n)))$.*

Proof. Given any structure \mathbf{A} (in the vocabulary of C_1) of size n , the corresponding structure $\Theta(\mathbf{A})$ has size at most $\theta(n)$. Let $k := \nu_{C_2}(\theta(n))$, then there is a formula ϕ in C^k for which it holds $\Theta(\mathbf{A}) \models \phi \Leftrightarrow \Theta(\mathbf{A}) \in C_2$. By composing ϕ with Θ , we obtain a formula ϕ' in C^{mk} that satisfies $\mathbf{A} \models \phi' \Leftrightarrow \mathbf{A} \in C_1$, where m is the width of Θ . This constant factor is accounted for in the O -notation. \square

2.4.3 Representation

In order to discuss definability of constraint satisfaction and optimization problems, we need to fix a representation of instances of these problems as relational structures. Here, we describe the representation we use, adapted from [4].

Numbers and Vectors. We represent an integer z as a relational structure in the following way. Let $z = s \cdot x$, with $s \in \{-1, 1\}$ being the sign of z , and $x \in \mathbb{N}$, and let $b \geq \lceil \log_2(x) \rceil$. We represent z as the structure \mathbf{z} with universe $\{1, \dots, b\}$ over the vocabulary $\tau_{\mathbb{Z}} = \{X, S, <\}$, where $<$ is interpreted the usual linear order on $\{1, \dots, b\}$; $S^{\mathbf{z}}$ is a unary relation where $S^{\mathbf{z}} = \emptyset$ indicates that $s = 1$, and $s = -1$ otherwise; and $X^{\mathbf{z}}$ is a unary relation that encodes the bit representation of x , i.e. $X^{\mathbf{z}} = \{k \in \{1, \dots, b\} \mid \text{BIT}(x, k) = 1\}$. In a similar vein, we represent a rational number $q = s \cdot \frac{x}{d}$ by a structure \mathbf{q} over the domain $\tau_{\mathbb{Q}} = \{X, D, S, <\}$, where the additional relation $D^{\mathbf{q}}$ encodes the binary representation of the denominator d in the same way as before.

In order to represent vectors and matrices over integers or rationals, we have multi-sorted universes. Let T be a non-empty set, and let v be a vector of integers indexed by T . We represent v as a structure \mathbf{v} with a two-sorted

universe with an index sort T , and bit sort $\{1, \dots, b\}$, where $b \geq \lceil \log_2(|m|) \rceil$, $m = \max_{t \in T} v_t$, over the vocabulary $(X, D, S, <)$. Now, the relation S is of arity 2, and $S^v(t, \cdot)$ encodes the sign of the integer v_t for $t \in T$. Similarly, X is a binary relation interpreted as $X^v = \{(t, k) \in T \times \{1, \dots, b\} \mid \text{BIT}(v_t, k) = 1\}$. In order to represent matrices $M \in \mathbb{Z}^{T_1 \times T_2}$, indexed by two sets T_1, T_2 , we have three-sorted universes with two sorts of index sets, or simply a single index set that consists of pairs. The generalization to rationals carries over from the numbers case. We write τ_{vec} to denote the vocabulary for vectors over \mathbb{Q} and τ_{mat} for the vocabulary for matrices over \mathbb{Q} .

Linear and Semidefinite Programs. We represent linear or semidefinite programs in their respective standard forms in the following way. An instance of a linear program in standard form is given by a constraint matrix $A \in \mathbb{Q}^{M \times V}$, and vectors $b \in \mathbb{Q}^M, c \in \mathbb{Q}^V$. Hence, we represent it as a structure over the vocabulary $\tau_{\text{LP}} = \tau_{\text{vec}} \dot{\cup} \tau_{\text{mat}}$.

Likewise, a semidefinite program in conic standard form is specified by a matrix-valued vector $\mathcal{A} \in \mathbb{Q}^{M \times (V \times V)}$, an objective matrix $C \in \mathbb{Q}^{V \times V}$, and a vector $b \in \mathbb{Q}^M$. This is again represented as a structure over $\tau_{\text{SDP}} = \tau_{\text{vec}} \dot{\cup} \tau_{\text{mat}}$. Sometimes it is more convenient to consider an SDP in inequality standard form, which is specified by a matrix $Z \in \mathbb{Q}^{M \times M}$, a matrix-valued vector $\mathcal{Y} \in \mathbb{Q}^{V \times (M \times M)}$ and an objective vector $c \in \mathbb{Q}^V$. Note that the vocabulary for both representations are the same, and that the conversion between the two standard forms can be expressed as an FPC interpretation, as it only involves simple substitution and rearranging of variables.

We can now state the definability result from [3], to the effect that there is an FPC interpretation that can define solutions to linear programs. We will show a generalization of the result to semidefinite programs in Theorem 22.

Theorem 14 (Theorem 11, [3]). *There is an FPC-interpretation Φ of $\tau_{\mathbb{Q}} \dot{\cup} \tau_{\text{vec}}$ in τ_{LP} that does the following:*

Let instances of a linear program be given by (A, b, c) with $A \in \mathbb{Q}^{M \times V}$, $b \in \mathbb{Q}^M$, and $c \in \mathbb{Q}^V$. Its feasible region is denoted by $\mathcal{F}_{A,b}$. Let \mathbf{I} be the relational representation of this LP.

Then, $\Phi(\mathbf{I})$ defines a relational representation of (f, v) , with $f \in \mathbb{Q}$, $v \in \mathbb{Q}^V$, such that

- *$f = 1$ if, and only if, $\max_{x \in \mathcal{F}_{A,b}} c^T x$ is unbounded;*
- *$v \notin \mathcal{F}_{A,b}$ if, and only if, there is no feasible solution;*
- *and $f = 0, v = \arg\max_{x \in \mathcal{F}_{A,b}} c^T x$ otherwise.*

CSPs. For a fixed domain D , and a constraint language Γ , we can represent an instance of $\text{CSP}(D, \Gamma)$ in a natural way. Namely, the vocabulary $\tau_{\text{CSP}(\Gamma)}$ consists of all relations in Γ . An instance $I = (V, C)$ is then represented as the τ_{Γ} -structure $\mathbf{I} = (V, (R^{\mathbf{I}})_{R \in \Gamma})$, where the universe is set to the set of variables V , and $s \in R^{\mathbf{I}}$ if there is a constraint $c = (s, R)$ in the constraint set C .

For the finite-valued variant, we define the vocabulary $\tau_{\text{VCSP}(\Gamma)}$ as $\tau_{\text{CSP}(\Gamma)} = \{(R_f)_{f \in \Gamma}, W, <\}$. An instance $I = (V, C)$ is then represented as a structure \mathbf{I}

with a three-sorted universe: A sort for variables V ; a sort of constraints C ; and a bit sort $\{1, \dots, b\}$ for some sufficiently large b . The relation $R_f^{\mathbf{I}} \subseteq V^{\text{ar}(f)} \times C$ then contains a tuple (s, c) if C contains a constraint of the form (s, f, w) . Similarly, the relation $W^{\mathbf{I}} \subseteq C \times 1, \dots, b$ encodes the weight of each constraint $c = (s, f, w)$ in the relational representation of integers, i.e. $W^{\mathbf{I}}(c, \cdot) = \{k \in \{1, \dots, b\} \mid \text{BIT}(w, k) = 1\}$. Finally, $<$ is again just interpreted as the usual natural order on $\{1, \dots, b\}$.

3 Main result

Our main result establishes a dichotomy for VCSP problems: Either $\text{VCSP}(D, \Gamma)$ is tractable, and every instance is captured by its basic linear programming relaxation; or there are instances that are only captured after $\Omega(n)$ levels of the Lasserre hierarchy, where n is the size of the instance. As a special case, we obtain the same dichotomy for the class of MAXCSP problems.

In the following, recall that we write $L_C(n)$ to denote the minimum number t , such that the Lasserre relaxation at level t suffices to capture all instances of C of size at most n , and we use ν_C to denote the counting width of a class C . For the sake of legibility, we use L_Γ as a shorthand for $L_{\text{VCSP}(D, \Gamma)}$ and ν_Γ as shorthand for $\nu_{\text{VCSP}(D, \Gamma)}$.

Theorem 15. *For any $\text{VCSP}(D, \Gamma)$ either every instance I is solved by $\text{BLP}(I)$; or $L_\Gamma(n) \in \Omega(n)$.*

Corollary 16. *For any $\text{MAXCSP}(D, \Gamma)$ either every instance I is solved by $\text{BLP}(I)$; or $L_{\text{MAXCSP}(D, \Gamma)}(n) \in \Omega(n)$.*

The key technical lemma here is a bound that relates the level of the Lasserre hierarchy required to capture all instances of a VCSP to the counting width of its class of decision problems.

Lemma 17. *For any $\text{VCSP}(D, \Gamma)$, it holds $L_\Gamma \in \Omega(\nu_\Gamma)$.*

In addition, we prove a counting width dichotomy for $\text{VCSP}(D, \Gamma)$. This is achieved by connecting the results of [9] and [5] to show a linear lower bound of ν_Γ for the hard cases of $\text{VCSP}(D, \Gamma)$.

Lemma 18. *If there are instances I of $\text{VCSP}(D, \Gamma)$ that are not solved by $\text{BLP}(I)$, then $\nu_\Gamma(n) \in \Omega(n)$.*

Given the above two lemmas, we obtain as a direct consequence Theorem 15. Hence we devote the remaining sections to proving Lemmas 17 and 18.

In Section 4 we provide a proof of Lemma 18. The main observation is that the relevant reductions described in [9] are essentially linear in size, and that we can reduce solving linear systems of equations over the two-element field to the hard cases of $\text{VCSP}(D, \Gamma)$. In turn, [5] shows that these systems of equations have linear counting width.

The general idea to prove Lemma 17 follows two main steps. In the first, we establish using Theorem 22 that the optimum value of a Lasserre SDP can be defined within FPC, given its explicit vector-matrix representation. This part will be proved in Section 5.

In Section 6, we then show that from any explicitly given 0–1 LP, we can define its explicit t -th level Lasserre relaxation by an FPC-interpretation using only $O(t)$ many variables. With the result of Section 5 this means that there is a FPC-formula in $O(t)$ variables that defines the solution to the given 0–1 program. On the other hand, if we know that the solution to some 0–1 program can not be defined using fewer than $\nu_\Gamma(n)$ many variables, then this implies a lower bound for the value of t of also $\Omega(\nu_\Gamma(n))$. This then concludes the proof of Lemma 17.

4 Counting width of finite-valued CSPs

In this section we aim to provide a proof for Lemma 18. The main pieces of the argument are known results from the literature, and we simply lay out how they together imply the claim.

We aim to show a linear lower bound for the counting width of those VCSPs that are not solved by the BLP relaxation. This aligns with the dichotomy result of Thapper and Živný (Theorem 3). That is, if $\text{VCSP}(D, \Gamma)$ is not solved by the BLP relaxation, we know that MAXCUT reduces to it. Our strategy is to show that (i) MAXCUT has linear counting width; and (ii) there is a linear size FPC-reduction from MAXCUT to $\text{VCSP}(D, \Gamma)$, if it is not solved by its BLP relaxation. By Proposition 13 this suffices to prove our claim.

For (i), we consider the problem 3LIN: An instance of 3LIN consists of a set of variables V , and two sets of equations, E_0 and E_1 . Each equation in E_0 has the form $a \oplus b \oplus c = 0$, where \oplus denotes addition modulo 2, and $a, b, c \in V$. Similarly, each equation in E_1 has the form $a \oplus b \oplus c = 1$. The problem is then to determine whether there is an assignment $h : V \rightarrow \{0, 1\}$ such that all equations are satisfied.

Lemma 19. $\nu_{3\text{LIN}}(n) \in \Omega(n)$.

Proof. In [5] Atserias et al. show a lower bound of for the counting width of the problem 3LIN that is proportional to the tree-width of the instance. More precisely, they show a construction that transforms any given graph $G = (V, E)$ with tree-width t into a pair of 3LIN instances (I, I') , each having $O(|V|)$ variables, such that I is satisfiable, but I' is not, and no C^k formula of at most t variables distinguishes between them.

The claim then follows by picking a class of graphs that have linear tree-width. Such graphs exist, for instance in the class of 3-regular expander graphs [1]. (A similar argument of picking linear tree-width graphs was already present in [6]). \square

As a direct consequence, we obtain that 3SAT also has linear counting width.

Lemma 20. $\nu_{3\text{SAT}}(n) \in \Omega(n)$.

Proof. Given an instance (V, E_0, E_1) of 3LIN, we replace each equation $a \oplus b \oplus c = 0$ by the four clauses containing a, b, c that have an even number of negated literals, i.e. $(a \vee b \vee c)$, $(\neg a \vee \neg b \vee c)$, $(a \vee \neg b \vee \neg c)$, and $(\neg a \vee b \vee \neg c)$. Similarly, each equation $a \oplus b \oplus c = 1$ is replaced by the four clauses of a, b, c that have an odd number of negated literals. This results in a 3SAT instance that is satisfiable if and only if the original 3LIN instance was satisfiable. Clearly, this is a linear size reduction that can be implemented in FPC. \square

We continue the reduction to MAXCUT.

Lemma 21. $\nu_{\text{MAXCUT}}(n) \in \Omega(n)$.

Proof. In [9], we find an explicit construction of a FPC-reduction from 3SAT to MAXCUT. This reduction is also linear size. \square

Finally, the reduction for (ii) has already been explicitly constructed in [9]. It is not difficult to confirm that these reductions are in fact linear in size. This chain of reductions then concludes the proof of Lemma 18.

5 Expressing semidefinite programs

We now turn to our definability result for semidefinite programs which states that the weak optimization problem for explicitly given SDPs is expressible in FPC. Our result relies heavily on previous work by Anderson et al. [3, 4] for the case of linear programming. In fact, their proof method allows a simple adaptation: The central piece there is a formulation of the ellipsoid method for polyhedra in FPC. That is, they show that the reduction from the optimization problem to the separation problem for polyhedra can be accomplished in FPC. They then show that the separation problem for explicit LPs is also definable in FPC. The same approach can be taken now for the case of SDPs, where we aim to solve the weak formulations of the optimization and separation problems. Formally, we prove the following.

Theorem 22. *There is an FPC-interpretation Φ of τ_{mat} in $\tau_{\text{SDP}} \dot{\cup} \tau_{\mathbb{Q}}$ that does the following:*

Let instances of a SDP be given by (A, b, C) and an error parameter δ , with $A \in \mathbb{Q}^{M \times (V \times V)}$, $b \in \mathbb{Q}^M$, $C \in \mathbb{Q}^{V \times V}$, and $\delta > 0$. Its feasible region is denoted by $\mathcal{F}_{A,b}$. Let \mathbf{I} be the relational representation of this SDP.

Then, $\Phi(\mathbf{I})$ defines a relational representation of $X \in \mathbb{Q}^{V \times V}$, such that

- *if $\mathcal{F}_{A,b}$ is empty or unbounded, there is no specification on X ;*
- *otherwise X is a δ -close and δ -maximal solution.*

5.1 Separation Oracle

Similar to the work in [3], our proof strategy is to use of an FPC-formulation of the ellipsoid method to reduce the optimization problem to the separation problem. Therefore in order to prove Theorem 22 we show first that we can express a separation oracle for SDPs in FPC.

Lemma 23. *There is an FPC-interpretation Φ of τ_{mat} in $\tau_{SDP} \dot{\cup} \tau_{\mathbb{Q}}$ that does the following:*

An instance of the separation problem is given by (A, b, Y) , and an error parameter δ , with $A \in \mathbb{Q}^{M \times (V \times V)}$, $b \in \mathbb{Q}^M$, $Y \in \mathbb{Q}^{V \times V}$, and $\delta > 0$. Let \mathbf{I} be the relational representation of this instance.

Then, $\Phi(\mathbf{I})$ defines a relational representation of $S \in \mathbb{Q}^{V \times V}$, such that

- *if $\mathcal{F}_{A,b}$ is empty or bounded, then there is no specification on S ;*
- *otherwise if $S = 0$, then Y is δ -close to $\mathcal{F}_{A,b}$;*
- *otherwise $\langle S, Y \rangle + \delta > \max\{\langle S, X \rangle \mid X \in \mathcal{F}_{A,b}\}$.*

Algorithm 1 describes a simple algorithm for the separation problem for the feasible region of SDPs. Its correctness follows from the fact that an infeasible point Y has to violate some inequality $\langle A_i, Y \rangle \leq b_i$ or the constraint $Y \succeq 0$. In the former case we can simply chose the separation normal as A_i , while in the latter case we can choose the normal to be $(-1)vv^T$ where v is an eigenvector corresponding to a negative eigenvalue λ of Y , since $\langle vv^T, Y \rangle = \lambda \cdot \|v\|^2$. In order to implement this algorithm in FPC however, we have to make two key modifications: (1) As we want our output to be a rational vector, we have to work with a finite precision in the calculations, and (2) all the steps must be definable in FPC. Together this leads to Algorithm 2 that solves the weak separation problem for SDPs, and is possible to translate to FPC using known techniques.

The translation of Algorithm 2 into an FPC-interpretation uses some known tools from descriptive complexity. First, we note that the basic vector and matrix operations, such as addition, multiplication, norm and even computing the characteristic polynomial can all be defined in FPC [12]. A key modification is in Line 4: In the original algorithm, we had to choose a violated constraint from an unordered set of constraints, which is in general not possible to express in FPC. However, we can employ the same technique as in [3]: the explicit choice of a constraint can be avoided by summing all violated constraints, since by linearity, the sum of violated constraints is again a violated constraint, which in turn is expressible in FPC.

In Line 6, we compute the eigenvalues of the input matrix Y up to a given precision $\delta/4$. This is possible in FPC since it is powerful enough to define the coefficients of the characteristic polynomial of definable matrices (see [12]).

Proposition 24. *There is an FPC interpretation of $\tau_{\mathbb{Q}}$ in $\tau_{mat} \dot{\cup} \tau_{\mathbb{Q}}$ that for a given a matrix $A \in \mathbb{Q}^{V \times V}$ and a value $\delta \in \mathbb{Q}$ (in their relational representation) defines the value of the smallest eigenvalue of A up to a precision of δ .*

Algorithm 2 Weak separation oracle for semidefinite programs

Input: $\mathcal{A} = \{A_1, \dots, A_m \in \mathbb{Q}^{V \times V}\}$, $b \in \mathbb{Q}^m$, $Y \in \mathbb{Q}^{V \times V}$, $\delta \in \mathbb{Q}$ such that $\delta > 0$.

Output: Solves weak separation problem on $\mathcal{F}_{\mathcal{A},b}$, Y , and δ .

```
1: function SEPARATION( $\mathcal{A}, b, Y, \delta$ ):
2:    $\mathcal{V} \leftarrow \{A_i \in \mathcal{A} \mid \langle A_i, Y \rangle > b_i\}$ 
3:   if  $\mathcal{V}$  is non-empty then
4:      $v \leftarrow \sum_{A_i \in \mathcal{V}} A_i$ 
5:     return  $\frac{v}{\|v\|}$ 
6:   Approximate eigenvalues  $\{\tilde{\lambda}_1, \dots, \tilde{\lambda}_{|V|}\}$  of  $Y$  up to precision  $\frac{\delta}{4}$ 
7:   if there is  $\tilde{\lambda}$  with  $\tilde{\lambda}_i < \frac{\delta}{2}$  then
8:      $v \leftarrow$  Vector satisfying  $\|(Y - \tilde{\lambda}_i I)v\| < \frac{\delta}{2}$  and  $\|v\| = 1$ 
9:     return  $(-1)/\|vv^T\| \cdot vv^T$ 
10:  return ACCEPT
```

Proof. Holm [12] establishes that there is an interpretation in FPC by which we can obtain from A the coefficients $\alpha_1, \dots, \alpha_n$ of the characteristic polynomial $p(x) = \det(xI - A) = x^n - \alpha_1 x^{n-1} + \dots + (-1)^n \alpha_n$. Since the coefficients have a linear order, by the Immerman-Vardi theorem, any polynomial time computable property can be defined in FPC, such as computing the smallest eigenvalue up to a precision δ . \square

Furthermore, the exact calculation of the eigenvector corresponding to a negative eigenvalue has been replaced by a linear optimization step in Line 8. In general, the eigenvectors corresponding to some eigenvalue λ are not uniquely defined. Not only can we scale eigenvectors by an arbitrary amount, in the case of an eigenvalue of higher multiplicity we have to choose a representative from a whole multidimensional eigenspace. To avoid this choice, we reformulate the problem as a linear program and rely on Theorem 14 to express this step in FPC. (While this LP-step is used as a blackbox here, in Section 5.2 we give some exposition on the techniques in [3] that are used to break the symmetry between choices.)

The correctness of the algorithm follows from some basic calculations. Assume the algorithm accepts an input $(\mathcal{A}, b, Y, \delta)$. Then Y violated none of the inequalities $\langle A_i, Y \rangle \leq b_i$, and all eigenvalues of Y are non-negative, and we can conclude that $Y \in \mathcal{F}_{\mathcal{A},b}$. Otherwise either some inequality $\langle A_i, Y \rangle \leq b_i$ is violated, in which case the algorithm produces a correct separating hyperplane, or some approximated eigenvalue $\tilde{\lambda}$ is smaller than $\delta/2$. In the latter case, the linear optimization step looks for a vector v such that $\|v\| = 1$ and $\|(Y - \tilde{\lambda})v\| < \delta/2$. Note that such a vector always exists. Let λ be the actual eigenvalue, with $\tilde{\lambda} = \lambda + \epsilon$ for some error ϵ with $|\epsilon| \leq \delta/4$. We have $(Y - \tilde{\lambda}I)v = (Y - \lambda I - \epsilon I)v$, and by setting v to some eigenvector corresponding to λ with $\|v\| = 1$, we get $\|(Y - \tilde{\lambda})v\| = \|(Y - \lambda I - \epsilon I)v\| = |\epsilon| < \delta/2$. Finally, given such a vector v , the normal of a weakly separating plane is given by $S := -1/\|vv^T\| \cdot vv^T$: Let

$\epsilon := Yv - \tilde{\lambda}v$. Then $\langle S, Y \rangle = -1/\|vv^T\| \cdot (v^T Y v) = -1/\|vv^T\| \cdot (v^T(\tilde{\lambda}v + \epsilon)) = -1/\|vv^T\|(\tilde{\lambda} + v^T \epsilon) < \delta$.

This shows that we can define a weak separation oracle for SDPs in FPC. For the next step, we show that the reduction from weak optimization to separation, i.e. the ellipsoid method, can be defined in FPC as well.

5.2 Reducing Optimization to Separation

In this section we construct a FPC-reduction from the weak optimization problem to the weak separation problem for SDPs.

Lemma 25. *If there is a FPC-interpretation expressing the weak separation problem for the feasible region of a given SDP, then there is a FPC-interpretation which expresses the weak semidefinite optimization problem.*

A version of this lemma was already proved in [3] but was stated in terms of the optimization and separation problems for polytopes. Their algorithm however generalizes nicely to our setting. Here, we give a brief overview of the main proof ideas again, and point out the changes we made to accommodate the case of SDPs. For a detailed description of their algorithm, we refer to [3, 4].

The main idea behind the construction is to repeatedly apply the separation oracle to define a linear order on the set of variables, and once a sufficient order is obtained, to apply the Immerman-Vardi theorem to define the ellipsoid method. This is achieved by defining a series of increasingly fine equivalence relations on the variable set V , specified by so-called *foldings* that we formalize below. Intuitively, these partitions are obtained as follows. In the beginning, every element of V resides in the same equivalence class. However, there may be some inputs on which the separation oracle returns a vector d with different values d_u and d_v for $u, v \in V$, which distinguishes the two elements u and v . In subsequent iterations, u and v are put in different equivalence classes, and this process is repeated until we obtain a sufficiently refined partition of V .

There are a couple of key modifications to be made to the algorithm from [3]. Namely, we show (1) that the folding operation preserves the positive semidefiniteness of sets, and (2) how to cope with the additional parameters introduced by the weak versions of the separation and optimization problems.

We start by defining the notion of *folding*.

Definition 26. *Let V be a non-empty set. For $k \leq |V|$, we call a surjective mapping $\sigma : V \rightarrow [k]$ an index map. Furthermore, for each $i \in [k]$ we define $V_i := \{v \in V \mid \sigma(v) = i\}$.*

For a vector $x \in \mathbb{Q}^V$, the almost-folded vector $[x]^{\tilde{\sigma}} \in \mathbb{Q}^k$ is given by

$$([x]^{\tilde{\sigma}})_i := \sum_{v \in V_i} x_v, \text{ for } i \in [k].$$

Its folded vector $[x]^\sigma \in \mathbb{Q}^k$ is given by

$$([x]^\sigma)_i := [x]^{\tilde{\sigma}}_i / |V_i|, \text{ for } i \in [k].$$

For a vector $\hat{x} \in \mathbb{Q}^k$, its unfolded vector $[\hat{x}]^{-\sigma} \in \mathbb{Q}^V$ is given by

$$([\hat{x}]^{-\sigma})_v := \hat{x}_i, \text{ with } v \in V_i, \text{ for all } v \in V.$$

For a given index map σ and a vector $x \in \mathbb{Q}^V$, we say x agrees with σ when for all $u, v \in V$ $\sigma(u) = \sigma(v)$ implies $x_u = x_v$. The notion also extends in a natural way to sets $S \subseteq \mathbb{Q}^V$, simply by defining the folded set $[S]^\sigma := \{[s]^\sigma \mid s \in S\}$. This can be seen as a projection of S into the (ordered) k -dimensional space \mathbb{Q}^k . When talking about matrices, that is, when the variable set V consists of pairs from some product set $V' \times V'$, we implicitly also require that σ is *consistent*. Namely we require that an index map $\sigma : V' \times V' \rightarrow [k] \times [k]$ is defined by an underlying index map $\tau : V' \rightarrow [k]$, with $\sigma(u, v) = (\tau(u), \tau(v))$.

There are some useful properties of the folding operation that allow us to infer some information about the geometry of a folded set from its original.

Proposition 27. *Let $\sigma : V \rightarrow [k]$ be an index map, x, c vectors in \mathbb{Q}^V , where c agrees with σ . Then,*

$$\langle c, [[x]^\sigma]^{-\sigma} \rangle = \langle c, x \rangle = \langle [c]^\sigma, [x]^\sigma \rangle.$$

Proof. See Proposition 12 from [3]. □

Proposition 28. *Let $\mathcal{P} \subseteq \mathbb{Q}^V$ be a polytope in \mathbb{Q}^V and let $\sigma : V \rightarrow [k]$ be an index map. Then the folded set $[\mathcal{P}]^\sigma$ is a polytope in \mathbb{Q}^k .*

Proof. See Proposition 13 from [3]. □

Proposition 29. *Let $X \in \mathbb{Q}^{V \times V}$ be a positive semidefinite matrix, and let $\sigma : V \times V \rightarrow [k] \times [k]$. Then $[X]^\sigma$ is also a positive semidefinite matrix in $\mathbb{Q}^{k \times k}$.*

Proof. Since we assume the index map σ to be consistent, we have a map $\tau : V \rightarrow [k]$ with $\sigma(u, v) = (\tau(u), \tau(v))$. Furthermore, since X is positive semidefinite, there exists vectors $g_v \in \mathbb{Q}^l$ for all $v \in V$ and some $l \geq 1$ such that $X_{u,v} = \langle g_u, g_v \rangle$ for all $u, v \in V$ (i.e. the Gram representation of X). Let us now define vectors $g_1^\tau, \dots, g_k^\tau$ by

$$g_i^\tau = \frac{1}{|V_i|} \sum_{v \in V_i} g_v,$$

where $V_i := \{v \in V \mid \tau(v) = i\}$. The vectors obtained in this way now form a Gram representation of the folded matrix $[X]^\sigma$, since

$$[X]_{i,j}^\sigma = \frac{1}{|V_{i,j}|} \sum_{(u,v) \in V_{i,j}} X_{u,v} = \frac{1}{|V_i||V_j|} \sum_{u \in V_i} \sum_{v \in V_j} \langle g_u, g_v \rangle = \langle g_i^\tau, g_j^\tau \rangle.$$

As the existence of a Gram representation implies positive semidefiniteness, this proves our claim. □

Since the feasible region of an SDP is the intersection of a polytope with the positive semidefinite cone, Propositions 28 and 29 show that the result of folding the feasible region of an SDP is again the feasible region of an SDP. Next we show that a weak separation oracle of the original set either serves as an oracle for the folded set, or produces some vector that does not agree with the index map of the folding.

Proposition 30. *Let $\mathcal{F} \subseteq \mathbb{Q}^V$ be a convex set, and let $\sigma : V \rightarrow [k]$ be an index map. Given a vector $x \in \mathbb{Q}^V$ that is δ -close to \mathcal{F} for some $\delta \geq 0$, the folded vector $[x]^\sigma \in \mathbb{Q}^k$ is also δ -close to the folded set $[\mathcal{F}]^\sigma$.*

Proof. Let $x = f + d$, where $f \in \mathcal{F}$ is some point in the set \mathcal{F} , and d the difference vector with $\|d\| \leq \delta$. By the definition of folding, we then have $[x]^\sigma = [f]^\sigma + [d]^\sigma$, where $[f]^\sigma$ is now a point in the folded set $[\mathcal{F}]^\sigma$. We can bound the norm of $[d]^\sigma$ by

$$\|[d]^\sigma\| = \sqrt{\sum_{i \in [k]} \left(\frac{1}{|V_i|} \sum_{v \in V_i} d_v \right)^2} \leq \sqrt{\sum_{i \in [k]} (\max_{v \in V_i} d_v)^2} \leq \sqrt{\sum_{v \in V} d_v^2} = \|d\|.$$

Since $\|d\| \leq \delta$, we have $\|[d]^\sigma\| \leq \delta$. \square

Proposition 31. *Let $\mathcal{F} \subseteq \mathbb{Q}^V$ be a convex set, and let $\sigma : V \rightarrow [k]$ be an index map. Given vectors $s, y \in \mathbb{Q}^V$ where s agrees with σ , and $\langle s, y \rangle + \delta > \max\{\langle s, x \rangle \mid x \in \mathcal{F}\}$, it holds that $\langle [s]^\sigma, [y]^\sigma \rangle + \delta > \max\{\langle [s]^\sigma, x \rangle \mid x \in [\mathcal{F}]^\sigma\}$.*

Proof. Let $x \in \mathcal{F}$ be a point in \mathcal{F} such that $\langle s, x \rangle$ is maximal. It follows from Proposition 27 that $[x]^\sigma$ is also a maximal point in $[\mathcal{F}]^\sigma$ with respect to $\langle [s]^\sigma, [x]^\sigma \rangle$. We then have

$$\langle [s]^\sigma, [x]^\sigma \rangle - \langle [s]^\sigma, [y]^\sigma \rangle = \langle [s]^\sigma, [x - y]^\sigma \rangle \leq \langle s, x - y \rangle = \langle s, x \rangle - \langle s, y \rangle < \delta.$$

For the first equality we use the fact that $[x - y]^\sigma = [x]^\sigma - [y]^\sigma$, and for the first inequality we use Proposition 27 to get $\langle [s]^\sigma, [x]^\sigma \rangle \leq \langle [s]^\sigma, [x]^\sigma \rangle = \langle s, x \rangle$. \square

Assume we are given a convex set $\mathcal{F} \subseteq \mathbb{Q}^V$ by means of a corresponding weak separation oracle, and some index map $\sigma : V \rightarrow [k]$. Proposition 30 ensures that whenever the oracle accepts some input (y, δ) , then the folded vector $[y]^\sigma$ is also δ -close to the folded set $[\mathcal{F}]^\sigma$. Likewise, by Proposition 31 we know that whenever the oracle for \mathcal{F} outputs a separation normal s that agrees with σ , then the folded vector $[s]^\sigma$ is also a δ -weak separation normal that separates $[y]^\sigma$ from $[\mathcal{F}]^\sigma$.

This leads us to a simple algorithm for the weak separation problem for $[\mathcal{F}]^\sigma$: On some input $\hat{y} \in \mathbb{Q}^k$ and $\delta \geq 0$, we simply give the input $[\hat{y}]^{-\sigma}$ and δ to the oracle for \mathcal{F} . If the oracle accepts, or returns a separation normal that agrees with σ , we are done. In the other cases, the oracle returns a separation normal that does not agree with our current index map σ , in which case we can use

that output to further refine the underlying equivalence relation. After at most $|V|$ such refinements we obtain a correct weak separation oracle for $[\mathcal{F}]^\sigma$.

For the overall algorithm, we can now follow exactly the procedure in [3], only substituting their blackbox for a separation oracle by the one we obtained from Section 5.1. To avoid duplication of the parts that stay unchanged, we refer to their work for the in-depth description of the algorithm, including the definition of the refinement procedure. This then concludes the proof for Lemma 25.

Together with the result from Section 5.1 that the weak separation oracle for the feasible region of an explicitly given SDP can be defined in FPC, this now almost establishes our main result of Theorem 22. A small technicality still remains: We assume as a condition in Theorem 22 that the feasible region of the given SDP instance is bounded and non-empty, while the original reduction given in Theorem 7 assumes the region to be bounded and full-dimensional. However, by means of a simple preprocessing step, a non-empty region can be turned into a full-dimensional one.

Assume we are given an SDP with a feasible region of $\mathcal{F} = \{X \in \mathbb{Q}^{V \times V} \mid X \succeq 0, \langle A_i, X \rangle \leq b_i, A_i \in \mathcal{A}, b_i \in b\}$, and we want to find a δ -close and δ -maximal point of $\mathcal{F}_{\mathcal{A}, b}$ with respect to some objective matrix C . We can then define an enlarged feasible region $\mathcal{F}' := \mathbb{Q}^{V \times V} \mid (X + \frac{\epsilon}{\sqrt{|V|}\|C\|}I) \succeq 0, \langle A_i, X \rangle \leq b_i + \frac{\epsilon}{\|A_i\|\|C\|}\}$ for some $\epsilon > 0$. Note that \mathcal{F}' is a non-empty, full-dimensional convex set, where every point is at most ϵ far away from the original set \mathcal{F} . Furthermore, it holds that $\max_{X \in \mathcal{F}'} \langle C, X \rangle \leq \max_{X \in \mathcal{F}} \langle C, X \rangle + \epsilon \langle C, X \rangle$. Hence, any δ -close and δ -maximal point of \mathcal{F}' is also a $\delta + \epsilon$ -close and $\delta + \epsilon$ -maximal point of \mathcal{F} . Consequently, by choosing ϵ sufficiently small, we can simply perform the optimization over the full-dimensional set, which is covered by Theorem 7.

This concludes the proof of Theorem 22. Note that the conditions on the feasible region of the definable SDP instances are readily satisfied for instance by those arising from finite-valued CSPs: The variables only range in $[0, 1]$, and there always exists a feasible solution. In fact, any (even non-optimal) assignment in the VCSP gives rise to a feasible solution of the 0–1 LP instance.

6 Lasserre lower bounds

We now apply the definability result on SDPs obtained in the previous section to prove Lemma 17.

The following proposition allows us to translate approximate solutions to exact ones. It quantifies the quality of approximation needed so that we can obtain the exact optimum of the original 0–1 problem by rounding an approximate optimum of its Lasserre SDP.

Proposition 32. *Let $I = (A, b, c)$ be a 0–1 linear program whose optimal solution is integral. Its feasible region is given by $\mathcal{K} = \{x \in \mathbb{Q}^V \mid Ax \geq b\} \cap \{0, 1\}^V$ with an objective vector $c \in \mathbb{Q}^V$. Furthermore let $\text{Las}_t^\pi(\mathcal{K}) = \mathcal{K}^*$ for some t ,*

and let $s \in \mathbb{Q}$ be the value of a $1/(4 \max\{1, \|c\|\})$ -close and $1/(4 \max\{1, \|c\|\})$ -maximal solution to $\text{Las}_t(\mathcal{K})$ under the objective c . Then, by rounding s , we obtain the exact optimal value for I .

Proof. Let s^* be the exact optimal value of $\text{Las}_t(\mathcal{K})$, and by assumption, also the optimal value for I . We argue that $|s - s^*| \leq 1/4$, and hence rounding s yields s^* , since $s^* \in \mathbb{Z}$.

First, note that the condition that s is $1/(4 \max\{1, \|c\|\})$ -maximal means that $s + 1/(4 \max\{1, \|c\|\}) \geq \max_{x \in \mathcal{K}} \langle c, x \rangle = s^*$. Hence, we have the lower bound $s \geq s^* - 1/4$.

The other direction follows from the fact that s is the value of a close solution, say, $s = \langle c, y \rangle$ for some $y \in \mathbb{Q}^V$. Since y is $1/(4 \max\{1, \|c\|\})$ -close to \mathcal{K}^* , it can be decomposed into $y = x + e$ where $x \in \mathcal{K}^*$ and $\|e\| \leq 1/(4 \max\{1, \|c\|\})$. The value of y is then bounded by $s = \langle c, y \rangle \leq \max_{x \in \mathcal{K}} \langle c, x \rangle + \langle c, e \rangle \leq s^* + 1/4$. \square

Next, we show that it is possible in FPC to define the t -th level of the Lasserre hierarchy for any explicitly given 0–1 program using only $O(t)$ many variables.

Lemma 33. *There is an FPC-interpretation from τ_{LP} to τ_{SDP} that for a given 0–1 linear program expresses the t -th level Lasserre hierarchy, using at most $O(t)$ many variables.*

Proof. Let an instance I of a 0–1 program be given by a matrix $A \in \mathbb{Q}^{U \times V}$, and vectors $b \in \mathbb{Q}^U, c \in \mathbb{Q}^V$. In order to show that we can define the t -th level Lasserre relaxation from I , it suffices to show that we can define the matrices $M_t(y)$ and $S_t^u(y)$ from I for any $y \in \mathbb{Q}^{\wp_{2t+1}(V)}$, $u \in U$. In particular, we represent $M_t(y)$ as a sequence of matrices $(\hat{M}_{t,q})_{q \in Q}$ where $Q := \wp_{2t+1}(V) \cup \{0\}$, such that $M_t(y) = \hat{M}_{t,0} + \sum_{q \in Q \setminus \{0\}} y_q \hat{M}_{t,q}$, and show that these matrices are definable. We represent $S_t^u(y)$ in an analogous way as $S_t^u(y) = \hat{S}_{t,0}^u + \sum_{q \in Q \setminus \{0\}} y_q \hat{S}_{t,q}^u$. Hence, here it suffices to argue that the matrices $(\hat{M}_{t,q})_{q \in Q}$ and $(\hat{S}_{t,q}^u)_{q \in Q}$ are definable from I within FPC.

Observe that for the t -th level Lasserre relaxation the matrices $M_t(y)$ and $S_t^u(y)$ are indexed by powersets $\wp_t(V)$, and the feasible region itself lies in a vectorspace indexed by $\wp_{2t+1}(V)$. As we need these index sets in our interpretation, we first describe how to define the powersets $\wp_k(V)$ for some fixed k . Namely, we encode a set $S \in \wp_k(V)$ by a k -ary tuple $T \in (V \cup \{0\})^k$ that contains each of the elements in S once, and where the symbol 0 fills the rest of the positions. As there are up to $k!$ many tuples encoding the same set, we additionally define an equivalence relation $\dot{=}_k$ on k -tuples that identifies two tuples if they are just permutations of each other. This can be defined by the following first order formulas.

$$\begin{aligned} \delta_{\wp_k}(x_1, \dots, x_k) &:= \bigwedge_{i \in [k]} (x_i = 0 \vee x_i \in V) \bigwedge_{i,j \in [k]} (x_i = 0 \vee x_i \neq x_j), \\ \dot{=}_k(x_1, \dots, x_k, y_1, \dots, y_k) &:= \bigvee_{\pi \in \text{Sym}(k)} \bigwedge_{i \in [k]} x_i = y_{\pi(i)}, \end{aligned}$$

where δ_{\wp_k} defines the set of tuples encoding some element in $\wp_k(V)$, and $\dot{=}_k$ defines a binary equivalence relation between those tuples. We use $Sym(k)$ to denote the set of permutations on $[k]$. From these definitions it is not hard to define basic set operations on the elements of $\wp_k(V)$. For instance, we can define a $4k$ -ary relation $union_k$ that encodes the union of two sets $S, T \in \wp_k(V)$.

$$union_k(x, s, t) = \bigwedge_{i \in [2k]} \left(x_i = 0 \vee \bigvee_{j \in [k]} x_i = s_j \vee x_i = t_j \right) \\ \bigwedge_{i \in [k]} \bigvee_{j \in [2k]} s_i = x_j \bigwedge_{i \in [k]} \bigvee_{j \in [2k]} t_i = x_j,$$

where $x \in \delta_{\wp_{2k}}$, and $s, t \in \delta_{\wp_k}$. Since $union_k(x, s, t)$ simply encodes $(x = s \cup t)$, we continue using the latter more familiar notation for set operations. One point to note here is that all formulas so far are all defined using $O(k)$ many variables.

Now we can turn to the definition of the matrices $(\hat{M}_{t,q})_{q \in Q}$ and $(\hat{S}_{t,q}^u)_{q \in Q}$. Each matrix $\hat{M}_{t,q}$ and $\hat{S}_{t,q}^u$ is indexed over the set $\delta_{\wp_t} \times \delta_{\wp_t}$. For $x, y \in \delta_{\wp_t}$ and $q \in \delta_{\wp_{2t+1}}$, their entries are given by

$$\hat{M}_t(q, x, y) = \begin{cases} 1 & \text{if } x \cup y = q \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\hat{S}_t^u(q, x, y) = \begin{cases} A_{u,v} & \text{if } \exists v \in V : x \cup y \cup \{v\} = q \\ -b_u & \text{if } x \cup y = q \\ 0 & \text{otherwise.} \end{cases}$$

By the above expressions $(\hat{M}_{t,q})_{q \in Q}$ and $(\hat{S}_{t,q}^u)_{q \in Q}$ are definable in FPC using only $O(t)$ variables. From this, we obtain the full SDP in inequality standard form by merging the constraints $M_t(y) \succeq 0$ and $S_t^u(y) \succeq 0$ into one single constraint of the form $Z \succeq 0$. □

Lemma 34. *Let D be a domain, and Γ a finite-valued constraint language. There is an FPC-interpretation of constant width from τ_Γ to τ_{LP} that defines for a given instance I of $VCSP(D, \Gamma)$ its corresponding 0–1 linear program.*

Proof. The 0–1 program that encodes a VCSP instance is given in Section 2.1. It has been shown in [9] that this LP is definable in FPC. The construction there also only uses a constant number of variables. □

Finally, we are now ready to prove Lemma 17.

Proof of Lemma 17. For the proof we fix a domain D and a finite-valued constraint language Γ . For better legibility, we write L_Γ for $L_{VCSP(D, \Gamma)}$, ν_Γ for $\nu_{VCSP(D, \Gamma)}$, and τ_Γ for the vocabulary $\tau_{VCSP(\Gamma)}$.

The proof idea is as follows. The argument is by contradiction. Suppose that $L_\Gamma(n) \in o(\nu_\Gamma(n))$. However, by composing the interpretations from Lemmas 34 and 33 and Theorem 22 we can define a formula ϕ that decides membership for the decision version of $\text{VCSP}(D, \Gamma)$ for instances of size n using only $o(\nu_\Gamma(n))$ many variables, which violates the assumed counting width bound of $\nu(n)$.

To be more precise, let Θ_t be the composition of the interpretations from Lemmas 34 and 33. That is, Θ_t is an interpretation of τ_{SDP} in τ_Γ that defines for a given VCSP instance $I = (V, C, w)$ the SDP of the t -th level of the Lasserre relaxation of the corresponding 0–1 linear program. Note that Θ_t is of width $O(t)$.

Note that the 0–1 linear programs corresponding to VCSP instances are always feasible, bounded in the 0–1 hypercube, and their optimum is always integral.

Suppose now $L_\Gamma(n) \in o(\nu_\Gamma(n))$, i.e. every instance $I = (V, C, w)$ of $\text{VCSP}(D, \Gamma)$ could be captured by some Lasserre relaxation of level $t \in o(\nu_\Gamma(|V|))$. Hence, $\Theta_t(I)$ defines a Lasserre relaxation whose optimal value is exactly the optimal value to I .

Then, by Theorem 22 there is an interpretation Σ of τ_{vec} in $\tau_{\text{SDP}} \dot{\cup} \tau_\mathbb{Q}$ that defines δ -close and δ -maximal solutions to $\Theta_t(I)$. Using Proposition 32, setting δ as $\delta = 1/4|C|$ allows us to obtain the exact optimal value for $\Theta_t(I)$ (and equivalently, for I) by means of rounding. Both defining the value for δ as well as the rounding can be done in FPC. Hence composing Θ_t and Σ , we obtain an interpretation Φ of width $O(t)$ that defines for a given instance of $\text{VCSP}(D, \Gamma)$ its optimal value.

Finally, using Φ it is not difficult to construct a FPC-formula ϕ using at most $O(t)$ many variables that decides membership for the decision version of $\text{VCSP}(D, \Gamma)$: For an instance (I, t) we simply compare $\Phi(I)$ to t . Since we assumed $t \in o(\nu_\Gamma(|V|))$, ϕ also uses only $o(\nu_\Gamma(|V|))$ many variables. This is a contradiction to the definition of ν_Γ . \square

7 Conclusion

We have established a dichotomy result, showing that *every* finite-valued CSP that is not solvable by its basic linear programming relaxation (and this includes all constraint maximization problems that are known to be NP-hard) requires a linear number of levels of the Lasserre hierarchy to solve exactly. Such linear lower bounds on the number of levels of the Lasserre hierarchy were known previously for specific CSPs. Our result shows that these are part of a sweepingly general pattern. This is established by considering the definability of semidefinite programs in logic, and using a measure of logical complexity, that we call *counting width*, to classify CSPs. This suggests some directions for further investigation.

A central motivating interest in semidefinite programming in general and Lasserre hierarchies in particular comes from their use in approximation algorithms. It would be interesting to extend our methods to show lower bounds

on the levels required to approximate a solution, as well as to obtain exact solutions. A potential direction is to define a measure of counting width, not just for a class of structures \mathcal{C} but based on the number of variables to separate two classes \mathcal{C}_1 and \mathcal{C}_2 . We could then seek to establish lower bounds on these numbers where \mathcal{C}_1 is a collection of instances of a VCSP with high optimum values and \mathcal{C}_2 contains only instances with low optima. This would show that instances with high optima cannot be separated from those with low optima by means of a small number of levels the Lasserre hierarchy.

Definability in FPC is closely linked to *symmetric* computation (see [2, 8]). In other words, algorithms that can be translated to this logic are symmetric in a precise sense. This suggests that many of our best approximation algorithms for constraint satisfaction, such as the Lasserre semidefinite programs are encountering a “symmetry barrier”. Breaking through this barrier, and coming up with algorithms that break symmetries, may be crucial to more effective approximation algorithms.

References

- [1] M. Ajtai. Recursive construction for 3-regular expanders. *Combinatorica*, 14(4):379–416, 1994.
- [2] M. Anderson and A. Dawar. On symmetric circuits and fixed-point logics. *Theory of Computing Systems*, pages 1–31, 2016.
- [3] M. Anderson, A. Dawar, and B. Holm. Maximum matching and linear programming in fixed-point logic with counting. In *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 173–182, 2013.
- [4] M. Anderson, A. Dawar, and B. Holm. Solving linear programs without breaking abstractions. *J. ACM*, 62(6):48:1–48:26, 2015.
- [5] A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666 – 1683, 2009.
- [6] J. Cai, M. Furer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. Technical report, Amherst, MA, USA, 1991.
- [7] E. Chlamtac and M. Tulsiani. *Convex Relaxations and Integrality Gaps*, pages 139–169. Springer US, Boston, MA, 2012.
- [8] A. Dawar. The nature and power of fixed-point logic with counting. *SIGLOG News*, 2(1):8–21, 2015.
- [9] A. Dawar and P. Wang. A definability dichotomy for finite valued CSPs. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015*, pages 60–77, 2015.

- [10] H-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 1999.
- [11] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.
- [12] B. Holm. *Descriptive Complexity Of Linear Algebra*. PhD thesis, University of Cambridge, 2010.
- [13] J. B. Lasserre. An explicit exact SDP relaxation for nonlinear 0–1 programs. In Karen Aardal and Bert Gerards, editors, *Integer Programming and Combinatorial Optimization*, volume 2081 of *Lecture Notes in Computer Science*, pages 293–303. Springer Berlin Heidelberg, 2001.
- [14] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.
- [15] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0—1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.
- [16] M. Otto. *Bounded Variable Logics and Counting — A Study in Finite Models*, volume 9 of *Lecture Notes in Logic*. Springer-Verlag, 1997.
- [17] T. Rothvoß. The Lasserre hierarchy in approximation algorithms. In *Lecture Notes for the MAPSP 2013 Tutorial*, 2013.
- [18] G. Schoenebeck. Linear level lasserre lower bounds for certain k-CSPs. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 593–602, 2008.
- [19] H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.
- [20] J. Thapper and Stanislav Ž. The complexity of finite-valued CSPs. In *Proceedings of the 45th ACM Symposium on the Theory of Computing*, STOC '13, pages 695–704, 2013.